Autonomy Lab: Autonomy Systems Prototyping

A Continuing Task Proposal for 632-07

Task Leaders

Elihu H. McMahon, JPL, elihu.h.mcmahon@jpl.nasa.gov, 818-354-4907, co-PI
Abdullah Aljabri, JPL, abdullah.s.aljabri@jpl.nasa.gov 818-354-5862, co-PI

Product Description

For FY 2000 the Autonomy Lab team will develop products that range from TRL 1 through 5. Three products at the system, subsystem and component level are planned, specifically:

1. Prototype Remote Agent software integrated with mission software which demonstrates autonomous execution of complex interferometer observations. This is an extension of the flight demonstrated Remote Agent for sequencing and fault protection of a large scale space-based interferometer instrument. This product will be used in the SIM and ST3 interferometry missions. Ames Research Center will provide tools for this product. **TRL 3 (FY2000), TRL 4 (FY2001), TRL 5 (FY2002)**.
2. Development of a prototype autonomous Attitude Control Subsystem using real-time embedded Java. We will pilot the use of the Java language for autonomous Guidance and control systems and address performance concerns which include: Basic Memory Management; Concurrency; Memory Mapped Input/Output; Timing Services; Performance (CPU time usage); Floating Point Math. **TRL 2 (FY2000), TRL 3 (FY2001), TRL 4 (FY2002)**
3. Explore the Java Operating system for the implementation of autonomous systems. Development of a "JavaGMC". The JavaOS will be ported to a generic microcontroller (GMC). Autonomy code for the control of a complex instrument will be ported to the JavaGMC and then tested for ease of implementation and performance. Perhaps in the future we could even fly the JavaGMC as an experiment. **TRL 1 (FY2000), TRL 2 (FY2001), TRL 3 (FY2002)**

State of the Art Technology	***Benefit for FY 2000***	***Benefit for FY 2001***
Remote Agent for Interferometry	Development of a Remote Agent prototype for ground-based interferometer instrument	Development of a Two-platform interferometer application of Remote Agent
ACS in Java	Estimated 20% reduction in time required for ACS code development	Estimated 50% reduction in time required for ACS code development
JavaGMC	Quantification of performance enhancement over Java Virtual Machine approach is the goal for this fiscal year	Native X2000 instrument code will be benchmarked

Continuing task. Push or Pull?
This is a **continuing** task. For the interferometer project this work is **50% push**, for mission projects ST3, ST4, OPSP the work is **100% push**. For technology-focused projects such as MDS the work is **33% push**.

Benefits

- Increased science return for interferometry missions is a high pay-off because the set-up of complex observations which require coordination of several telescopes is accomplished more efficiently with an autonomous system.
- Increased mission robustness. Another strength of the Remote Agent is autonomous planning and reconfigurability in the face of faults to continue science data acquisition.
- The Autonomy Lab is particularly suited for this research because it provides:
  - A complete end-to-end testbed with integrated flight and ground software components on a flight-like processor a unique capability at JPL outside of flight projects.
  - It is staffed by technologists and mission developers for the express purpose of inventing and maturing technologies that promote autonomous systems and inserting them into missions.
  - It pioneers the use of emerging space hardware such as processors (PPC750), data transmission buses (IEE 1394, I2C), and architectures that examine multiple processors utilization.
- ACS in Java and the Java GMC are critical to understanding the issues and tradeoffs of the development of complex autonomy software written in Java. The JavaGMC task would give us an extremely good set of benchmarks with which to compare Embedded Real-Time Java to the baseline traditional implementation, and will offer a significant performance advantage by eliminating the expensive context switching between the Java Virtual Machine and VxWorks. It could very well offer comparable performance to native C or C++ code running directly on top of the real-time kernel. The JavaGMC will be the pathfinder which will allow NASA to be the first to fly Java applications for space robotics.

The largest payoff of these products is the systemized reduction of the risk of technology insertion in the autonomy lab and thus gaining the acceptance of project managers and the mission development community. Flight implementation will in turn help realize the broader benefits of autonomy technologies such as enabling new exploration, reducing development cost through reuse of models, providing mission robustness through better real-time response, and reducing operation costs by migrating functions to flight.

Technical Approach

Remote Agent for Interferometer Missions

The Remote Agent configuration on VxWorks will be developed for sequencing and fault protection of a large scale space-based interferometer instrument. The Remote Agent is an autonomous on-board agent that can perform complex tasks which would otherwise require human sequencing and intervention. It uses reasoning to determine the proper course of action based upon its experiences and models of the environment. The models

are provided by engineers for the specific mission to be performed. The Remote Agent technology has been successfully demonstrated on the Deep Space One mission.

The process of obtaining a high-resolution image from an interferometer is complex. It requires that the instrument be oriented in several directions in a plane perpendicular to the line-of-sight to the target. It also requires that the telescopes be spaced at different distances (called 'baselines') for each orientation. The SIM mission will accomplish this by having several telescopes which can be paired together in various combinations. The ST3 mission will fly one telescope on each of two separate spacecraft and vary the distance between the two spacecraft.

Using the Remote Agent will improve efficiency and reduce mission costs by allowing the spacecraft to perform the complex task of obtaining interferometric observations with only high-level guidance from ground controllers. The spacecraft will determine on-board when a sample is complete and then command the next appropriate sample. The Remote Agent will also be able to detect on-board equipment faults and reconfigure the system to continue observing, if possible. These capabilities will enable the maximum science return during the mission life.

The following steps shall be accomplished in order to configure and adapt the Remote Agent for use on SIM and ST3:

1. The models upon which Remote Agent reasoning is based shall be updated in general to perform the task of interferometry.
2. The Remote Agent itself shall be ported to the hardware and software environment being baselined for each of the two missions.
3. The general interferometry models shall be tailored to the needs of each specific mission.
4. A CORBA implementation (ACE/TAO) will be used for module interfaces.
5. The Remote Agent shall be tested and validated for use in this new role.

These steps are not necessarily sequential. For example, testing and validation will surely occur at each step, and the steps are envisioned to be more cyclic in nature. Initially, the models will be general, but over time they will be refined as the system becomes more concrete and the design details emerge.

Java Attitude Control System

The next generation of spacecraft will attempt to take advantage of modern software development tools and methodologies. The primary methodology used in modern software development is *Object Oriented Design* (OOD). Although there are many programming languages that support OOD (such as Ada, C++, Smalltalk, etc), in recent years one language has emerged as the language of choice in commercial software development - that language is *Java*.

Although there are many advantages to using Java, this language was not designed with consideration for the problems of embedded real-time systems. The specific features of Java that are of concern to developers of embedded real-time systems include:

1. The strategy adopted for Garbage Collection (GC). Can GC be scheduled for a time that won't interfere with deterministic timing? Can objects be manually identified, or given a priority for collection?
2. Java threads. Are Java threads mapped directly to native Operating System (OS) tasks? Specifically, how is this done in a popular embedded real-time OS such as VxWorks? Can Java threads make use of VxWorks semaphores, task scheduling, and message queues?
3. Floating point numbers. Java currently only implements a subset of the IEEE Standard 754, which defines standard binary floating-point arithmetic operations. Floating point calculations in Java exhibit non-standard characteristics, such as the way rounding is performed. What design implications does this precipitate for software that depends upon heavy floating point number crunching?
4. CPU usage. Since Java is an interpreted language running in a virtual machine, what is the performance compared to other OOD languages such as C++? Are Java just-in-time byte-code compilers appropriate for real-time systems?
5. Memory usage. How big are the Java Virtual Machine (VM) and class libraries? Can they be scaled by removing unused features and classes? How much dynamically allocated memory does a typical Java embedded real-time application use?

Sun Microsystems is beginning to address the needs of the embedded real-time market, and changes to Java are being considered to respond to these types of concerns. This task is part of a joint JPL/Sun prototyping effort to build a spacecraft embedded real-time system in Java. Specifically, this work is being done in support of the JPL Mission Data System (MDS) assessment of embedded real-time Java. The prototype Java Attitude Control System (ACS) would query sensors, execute floating point control algorithms, and issue control commands in a deterministic, real-time context. It would be developed for an embedded real-time OS (such as VxWorks), running on a modern embedded CPU architecture (such as PowerPC). The development of such a prototype would generate a better understanding of the above issues as they relate to flight software design and implementation.

Nilsen, K. Issues in the Design and Implementation of Real-Time Java. 1999, http://www.sys-con.com/java/iss1/real.htm

Sun Microsystems Inc. The Java Language Environment: A White Paper. 1996, Sun Microsystems, Inc.: Mountain View, CA.

Nilsen, K. Real-Time is No Longer a Small Specialized Niche, in Fifth Workshop on Hot Topics in Operating Systems (HotOS-V). 1995. Orcas Island, Washington: IEEE Computer Society Press.

JavaGMC

In most typical embedded environments, Java applications run completely within a Java Virtual Machine (JVM), which is itself an application running within the context of a host operating system. As such, it is unreasonable to think that the performance of Java applications will ever approach the performance of "native" applications written specifically for the host operating system.

In order to provide developers with all the benefits of the Java language, yet overcome the inherent performance limitations of the JVM running on top of a host OS, a few companies have developed a

complete operating system designed specifically to run Java applications directly on the host hardware – JavaOS.

The X2000 Generic Micro-Controller (GMC) is a small, lightweight computing element designed to act as an instrument interface to the rest of the X2000 spacecraft. In the "baseline" GMC, Java applications run within the VxWorks JVM.

The end product of the proposed research will be a version of the JavaOS ported directly to an X2000 GMC. Autonomy code for the control of a complex instrument will be ported to the JavaGMC and will be used to provide a series of benchmarks comparing the baseline approach to the pure-Java approach. The JavaGMC will coexist on the X2000 high-speed data bus with possibly dozens of other computing nodes.

In order to successfully complete the proposed project, it will be necessary to perform the following steps:

1. *Partner with X2000 to acquire a GMC.*
2. Partner with a third-party developer, such as Sun Microsystems, to develop the GMC JavaOS implementation.
3. Identify and partner with one of the JPL instrument sections to port their instrument code to Java (if they have not already done so).
4. Develop a series of benchmarking tests to evaluate the results.
5. Publish a white-paper on the findings of the research.

White papers on JavaOS are available online from Sun Microsystems (http://wwwwswest2.sun.com:80/javaos/) and IBM (http://www.ibm.com), and Intel (http://www.intel.com).

Evaluation Factors: Our approach addresses the following selection criteria:

Innovation – explores new language (Java), new hardware and new interface protocols for implementing autonomy.

State-of -the -art – unique intersection of autonomy technology, emerging software infrastructure/hardware and advance mission applications.

Feasibility – ensured by multi-discipline team and variable fidelity (up-to mission level) testbed tools.

Significance – working prototype products demonstrate quantitatively benefits of autonomy to clarify significance

Track record – team obtained 1999JPL award of excellence for conceiving, implementing and flight demonstration of autonomous station keeping on TOPEX/Poseidon mission.

Status and Milestones

1. In FY1999 the Autonomy Lab developed and established a unique end-to-end testbed environment that includes a spacecraft simulator, flight software, a ground data system with flight-like commanding and telemetry, and visualization tools under this task.
2. We have established customers, partners and co-developers who include; Ames Research Center, projects including X2000, MDS, SIM and ST3.
3. Under the above partnerships the autonomy lab has established a co-funding arrangement for prototyping work. Accomplishments in FY1999 under this arrangement include:
   - Development of a CORBA/ACE TAO interface for mission software standardization and communication. ($18K: Alab, $55K: MDS)

- Development of IP on 1394 that enables the transmission of TCP/IP datagrams between multiple computers connected by an IEEE 1394 Serial Bus (FireWire). *Never been done before*. ($14K: A-lab, $23K: TMOD)
- Development of real-time embedded mission software in the Java language. Deep Space One flight software was rewritten in Java to characterize performance. ($8K: Alab, $27K: MDS)

FY 2000 Milestones, Deliverables:
- A Remote Agent prototype for ground-based interferometer instrument.
- A prototype Attitude Control Subsystem using real-time embedded Java for cruise phase.
- Assessment of Java OS versus Java on a virtual machine.

FY 2001 Milestones:
- Two-platform interferometer application of Remote Agent.
- Guidance and Control Subsystem using real-time embedded Java for rendezvous and docking.
- Prototype Java OS implementation.

FY 2002 Milestones:
- Multi-platform interferometer application of Remote Agent
- Guidance and Control Subsystem using real-time embedded Java for landing and ascent.
- A "JavaGMC" instrument.

Customer Relevance

This task supports all three enterprises since they utilize autonomous flight systems.

Space Science Enterprise (SSE)
- ORIGINS - Space Observatories
- Spacecraft control, formation flying beam combining and fringe stabilization.
- SOLAR SYSTEM EXPLORATION
- Robust economical colonies of intelligent robots for planetary exploration.
- STRUCTURE AND EVOLUTION OF THE UNIVERSE
- Large ultra-light, ultra-stable observatory structures, high-precision formation flying for very long baseline interferometry.

Earth Science Enterprise (ESE)
- Protocols, strategies, and spacecraft technologies that enable the delivery of space-acquired data, control of spacecraft, or hybrid network management, through the appropriate and cost-effective use of existing and planned commercial space and terrestrial communications infrastructures.
- Fault detection, isolation, and recovery; self-repairing systems and architectures; automated model-based reasoning.

Human Exploration and Development of Space (HEDS)
- Autonomous cognitive systems for rapid problem diagnosis and resolution, medical diagnosis and experiment management

- Aero-capture; aero-entry; precision landing; hazard avoidance and autonomous rendezvous and docking.
- Autonomous and semi-autonomous robotic systems for in space maintenance and repair, and surface activities including maintenance and repair, science, asset deployment and construction, and crew assistance.

Letters of Support

ACS in Java and the Java GMC are critical in understanding the issues and tradeoffs of Mission Data System's goal to fly flight software written in Java. Letters of support are forthcoming from Sandy Krasner and Allan Sacks (MDS).

The Remote Agent technology will be used in interferometry missions such as SIM and ST3. Letters of support are forthcoming from Tom Lockhart (ST3/RICST) and Guy Man (NMP).

Qualifications

The Autonomy Lab proposal team has over 25 years of combined experience in embedded real-time programming. We have experience as authors of portions of the Deep Space One flight and simulation software, and thus, are familiar with the Remote Agent. We are currently in the process of integrating the Remote Agent into interferometry software. We are also leading a pilot task to characterize the performance of Java for real-time applications. We have successfully ported VxWorks BSP real-time operating system to custom PPC hardware (similar to GMC) and have extensive experience with PPC-based embedded systems and real-time kernels. Candidate for NASA Software of the Year 1998 for work done in Java (winner not announced yet).

June 29, 1999

Autonomy and Information Management JPT (632-07)

Subject: Autonomy Lab: Autonomy Systems Prototyping

Reference: Letter of Support

The purpose of this letter is to indicate my support for the work in development and prototyping of autonomous systems in the Autonomy Lab led by Elihu McMahon and Abdullah Aljabri and funded by the Autonomy and Information Management JPT (632-07).

Mission autonomy technology infusion represents a major goal to achieve automation of mission operations required for low-cost, reliable future missions. As such, these technologies are central to the autonomy goals of JPL's Mission Data System.

Current plans for the Autonomy Lab to prototype autonomy for missions are closely aligned with the Mission Data System's goals and are relevant to both Deep Space and Earth Orbiting missions (and hence, both NASA's Space Science and Mission to Planet Earth enterprises). Thus, I expect that this work will significantly contribute to the Mission Data System development. Therefore, I strongly support funding for the Autonomy Lab prototyping tasks.

Sanford Krasner
Lead, Mission Data System Guidance, Navigation and Control

June 28, 1999

Autonomy and Information Management JPT (632-07)

Subject: Autonomy Lab: Autonomy Systems Prototyping

Reference: Letter of Support

The purpose of this letter is to indicate my support for the work in development and prototyping of autonomous systems in the Autonomy Lab led by Elihu McMahon and Abdullah Aljabri and funded by the Autonomy and Information Management JPT (632-07).

The Remote Agent is important to these missions because each of them require an integrated systems solution to problems requiring autonomous, real-time decision-making. Interferometers rely on precise, highly coupled operation of complex electro-optical mechanical subsystems. These systems cannot be fully characterized on the ground before launch and require regular alignment and calibration once in space. The Remote Agent's value to these systems is its ability to take real-time, situation-based adaptive action to maintain interferometer operation in the presence of changing instrument performance. Without Remote Agent, it is expected that much of the mission's life span would be taken up by ground-in-the-loop, labor-intensive recovery activities that would severely limit the mission's science return.

A second invaluable feature of the Remote Agent is the ability to modify sequences at runtime without rebooting. A developer working in a testbed environment can work with a new sequence and fix bugs in it without having to reboot and recompile. This is critical in the development of complex sequences, where it can often take many minutes or even hours to reach a particular state.

In conclusion, before the Remote Agent was available there were no solutions to many of the autonomy issues that are inherent in the complex work of interferometry. The Remote Agent is a developed and demonstrated technology needed to make autonomous interferometry viable and it is the baseline autonomy tool for the Keck Interferometer, the ST-3 interferometer, and the SIM instrument. Remote Agent control of practical interferometer operations scenarios has been demonstrated on SIM testbeds and the technology is partly being infused into interferometry through the efforts supported by the Autonomy Lab under task (632-07).

Thomas Lockhart
Place Title Here

June 29, 1999

Autonomy and Information Management JPT (632-07)

Subject: Autonomy Lab: Autonomy Systems Prototyping

Reference: Letter of Support

The purpose of this letter is to indicate my support for the work in development and prototyping of autonomous systems in the Autonomy Lab led by Elihu McMahon and Abdullah Aljabri and funded by the Autonomy and Information Management JPT (632-07).

Mission autonomy technology infusion represents a major goal to achieve automation of mission operations required for low-cost, reliable future missions. As such, these technologies are central to the autonomy goals of NASA's New Millennium Program.

Current plans for the Autonomy Lab to prototype autonomy for interferometry missions are closely aligned with the New Millennium Program's goals and are relevant to both Deep Space and Earth Orbiting missions (and hence, both NASA's Space Science and Mission to Planet Earth enterprises). Thus, I expect that this work will significantly contribute to the New Millennium Program. Therefore, I strongly support funding for the Autonomy Lab prototyping tasks.

Guy K. Man
Lead, New Millennium Program IPDT

June 29, 1999

Autonomy and Information Management JPT (632-07)

Subject: Autonomy Lab: Autonomy Systems Prototyping

Reference: Letter of Support

The purpose of this letter is to indicate my support for the work in development and prototyping of autonomous systems in the Autonomy Lab led by Elihu McMahon and Abdullah Aljabri and funded by the Autonomy and Information Management JPT (632-07).

Mission autonomy technology infusion represents a major goal to achieve automation of mission operations required for low-cost, reliable future missions. As such, these technologies are central to the autonomy goals of JPL's Mission Data System.

Current plans for the Autonomy Lab to prototype autonomy for missions are closely aligned with the Mission Data System's goals and are relevant to both Deep Space and Earth Orbiting missions (and hence, both NASA's Space Science and Mission to Planet Earth enterprises). Thus, I expect that this work will significantly contribute to the Mission Data System development. Therefore, I strongly support funding for the Autonomy Lab prototyping tasks.

Allan Sacks
Manager, Mission Data System